

# Experimental Performance Evaluation of WebRTC Video Services over Mobile Networks

Mohamed Moulay  
IMDEA Networks Institute, Madrid, Spain  
and University Carlos III of Madrid, Spain  
mohamed.moulay@imdea.org

Vincenzo Mancuso  
IMDEA Networks Institute, Madrid, Spain  
vincenzo.mancuso@imdea.org

**Abstract**—With the rapid growing market for smartphones, and user’s confidence for immediate access to high-quality multimedia content, the delivery of video over wireless networks has become a big challenge. WebRTC is a new and evolving standard that has been developed specifically to meet this demand and enable a high-quality experience for mobile users of real time communication services. However, little systematic experimental studies have been carried out so far to assess the service experienced by users in a realistic mobile setting. In this work, we describe measurements collected from a WebRTC implementation operated from real mobile nodes within the pan-European MONROE platform. Using data from application and network, in different wireless environments, we experimentally investigate and compare the performance of WebRTC for static and mobile cellular users of several networks across Europe.

**Index Terms**—WebRTC; MBB; Experiments.

## I. INTRODUCTION

Web Real-Time Communication (WebRTC) proposes to easily integrate video services in web browsers, based on local tools [1]. In turn, such tools are based on well-known web technologies, able to simply integrate audio, video, and data transfer operations of the real-time communication protocol (RTC) into a normal webpage. The WebRTC project<sup>1</sup> was first introduced by Google as an open source project, and then other software developers and telecom vendors joined, which has led to integration of WebRTC into commercial browsers like Chrome, Opera and Firefox [2], [3].

Since offering video services and multimedia channels is a *killer application* for Mobile broadband (MBB) networks, WebRTC and similar projects impose stringent quality requirements on such networks, that are nowadays evolving from 4G to 5G under the pressure of a steadily increasing number of mobile users [4]. Therefore, there is now a strong need for objective information about MBB performance and reliability to support video and multimedia mobile services. Thus, various initiatives have arisen, among which the US FCC’s Measuring Broadband America initiative [4] and MONROE [5], to monitor and assess MBB performance.

We focus on WebRTC performance figures in mobile environments, for which so far little experimental work exists. In fact, other works on assessing WebRTC performance figures are currently sprouting, but they have so far only investigated

basic properties in controlled environments. For instance, the authors of [6] used a cloud-engineered automatic testing tool for WebRTC, although they have not tested the service offered by mobile operators and core networks. Similarly, the authors of [7] have experimentally tested one-to-many communications over WebRTC (namely “simulcast”), although their experiments are limited to a gigabit LAN environment.

In contrast, for our work, we leverage on a large-scale on-line measurement platform and focus on users connecting through MBB networks only. Specifically, we use the above mentioned MONROE platform, which has been designed and is currently operated in the frame of a European project aimed at providing multi-homed, independent, large-scale monitoring and evaluation of performance for mobile broadband networks in heterogeneous environments. Acquiring access to this platform allows for the deployment of vast measurement setups to collect data from operational MBB networks in various European countries. Differently from other approaches based on operator-driven quality-assessment campaigns [8], [9], or on traditional drive-by tests [10], MONROE offers an open platform for repeatable and traceable experiments. Besides, it offers open access to collected data, which refer to multiple operators, and includes device-level metadata, which is the key to use and possibly filter results without raising user’s privacy concerns. Therefore, this platform offers much richer data than what can be offered by crowdsourcing initiatives like, e.g., Netalyzer [11] and Haystack [12].

In this paper, we use the MONROE platform to investigate session-related performance statistics linked to the use of an off-the-shelf, WebRTC-based streaming application. This application enables streaming videos in real time with high quality using web browsers that support WebRTC (e.g., Chrome, Firefox) [1]. When using Google Chrome, data from both the sending and receiving parties in a WebRTC-based *telemeeting*, can be gathered via the WebRTC internals page,<sup>2</sup> thus making it possible to get a complete overview of the stream in the mobile nodes. Such session-related statistics may help to identify root causes and track the origins of performance issues in video streaming, so to understand how these technical factors impact Quality of Service (QoS) offered by the network and Quality of Experience (QoE) enjoyed by

<sup>1</sup><http://www.webrtc.org/>

<sup>2</sup><chrome://webrtc-internals/>

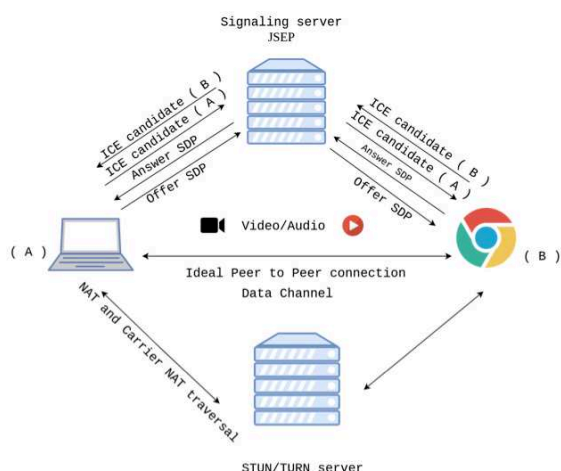


Fig. 1: WebRTC peer-to-peer communication.

the users. Indeed, gathering such insights is crucial and may steer the development of real-time communication schemes and intelligent optimization strategies. Our results show that current European MBB networks provide static users with enough resources and QoS to suitably make use of WebRTC, whereas mobility dramatically worsens network performance, often resulting in unacceptably low levels of QoE.

In the remainder of the paper, we start by providing background on WebRTC in Section II. We present the MONROE platform in Section III and our measurement setup in Section IV. Section V focuses on experimental results. Eventually, Section VI summarizes the findings of the paper.

## II. WEBRTC

### A. Real-time communications to and from browsers

The initial idea behind the development of a mechanism for web-based real-time communication like WebRTC or other proposals discussed in standardization fora was mainly to provide a tool to empower web browsers and make multimedia communications easier than before [1]. Specifically, the WebRTC approach is based on well-known web technologies like HTML and JavaScript, which are able to simply integrate RTC into a webpage.

Supported by Google first, and then by other web browser developers (Firefox, Opera) and telecom vendors (Ericsson, Cisco, Alcatel-Lucent) [2], WebRTC integrates video streaming capabilities into web browsers without the need of installing plugins or third-party software. Its standardization—led by W3C and IETF—helps separating application and communication level duties of video services [13]. In particular W3C and IETF are jointly concentrated on defining JavaScript Application Programming Interfaces (APIs) and a peer-to-peer communication mechanisms between browsers, to allow direct and possibly server-less video communication [14]. The designed API for WebRTC is capable of managing a browser based client-side RTC with host-to-browser connection, browser-to-browser connection management, encoding/decoding, NAT traversal and media streaming [14].

The main API highlights are as follow:

- **getUserMedia**: it provides with agile access to user media such as microphones, cameras and display.
- **RTCDataChannel**: it authorizes data transfer through peer-to-peer channels.
- **RTCPeerConnection**: it is responsible for setting up a direct connections between two WebRTC applications, which allows data channels and media streams to be carried.

This API represents the base of any WebRTC application. Besides, it is possible to add plugins to exploit WebRTC or to empower it with, e.g., encryption and security features [15].

### B. Protocols and Communication Services

Web browsers have to support several communication protocols and communication services to enable WebRTC. As shown in Fig. 1, this includes mechanisms such as signaling, session establishment, transport and security.

For transport, WebRTC commonly uses UDP with DTLS security, which is a TLS extension for unreliable datagram transport. However, WebRTC gives the option to use TCP with TLS as well. In addition, it uses the Stream Control Transport Protocol (SCTP) and the Secure Real-Time Transport (SRTP) to control media channels and handle encryption keys. SCTP is capable of multiplexing multiple application data streams and provides reliable delivery of UDP datagrams. Hence, a peer-to-peer secured media path can be established by leveraging SCTP and SRTP.

For what concerns the session setup and the negotiation of media features and connection parameters, WebRTC uses SIP or XMPP with parameters conveyed through the Session Description Protocol (SDP). In addition, WebRTC uses the JavaScript Session Establishment Protocol (JSEP) as session setup mechanism, which endorses secured signaling, and encrypted data transport over either DTLS/UDP or TLS/TCP. It also uses signaling servers that help initiate peer-to-peer multimedia capabilities handshaking and establish connections.

WebRTC also includes mechanisms for firewall and NAT traversal. Specifically, it inherits from the VoIP world the Interactive Connectivity Establishment (ICE) framework. ICE helps to use TURN/STUN, which in turn eases WebRTC peers to test and collect preferred “candidates of communication” options, i.e., it sorts the known transport addresses of a media destination to which is possible to attempt connection. Indeed, carefully selected communication candidates are the key to maximize the chance of success.

The most recent specifications on WebRTC have been released in November 2017 with the W3C EditorDraft “WebRTC 1.0: Real-time Communication Between Browsers”.<sup>3</sup>

## III. THE MONROE MEASUREMENT PLATFORM

Since we use MONROE for our WebRTC measurements, here we present a short overview on the measurement platform. The main idea behind MONROE is to deploy hundreds

<sup>3</sup><https://www.w3.org/TR/webrtc/>

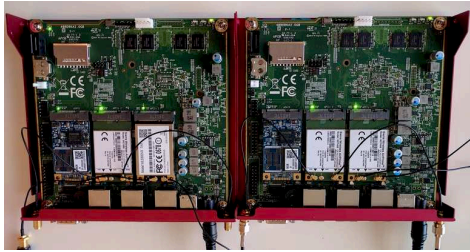


Fig. 2: A MONROE node pair with three LTE Cat6 modems and one WiFi adapter.

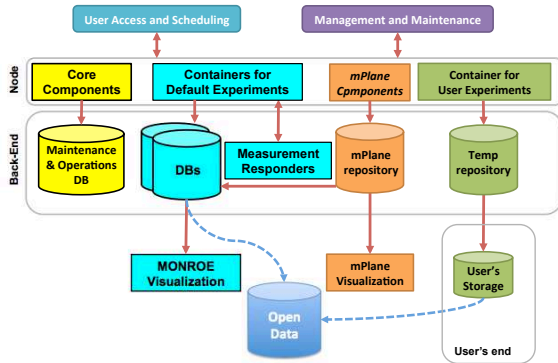


Fig. 3: MONROE platform components and information flow.

of measurement points (namely, MONROE nodes) across Europe to support multiple research oriented experiments and to collect, analyze, visualize and share the measurements. Nodes are distributed in different places such as in labs, research institutes, universities, buses, trucks, and trains across Spain, Italy, Norway, and Sweden. Thus, MONROE nodes operate as MBB static or mobile users, with their off-the-shelf or custom applications. Fig. 2 shows the hardware of a MONROE node, which consists of a dual mini-motherboard system, using apu2 boards.<sup>4</sup> One of the two boards, namely the one in the “head” node, mounts two Sierra Wireless MC7455 miniPCI express modems,<sup>5</sup> while the other board (in the “tail” node) has one MC7455 modem and a WiFi card. The modems are 4G devices supporting the most recent LTE category an industry-grade modem could provide at the time of node design (CAT6). More details on the node hardware and design can be found in [5].

The platform is open and meant to provide experiments as a service, so that all hardware specifications, open software, user manuals and experiment templates are made available on github,<sup>6</sup> while scientific publications and data reports are accessible and searchable from the MONROE webpage.<sup>7</sup>

#### A. MONROE Components

The MONROE architecture is illustrated in Fig. 3 and consists of several blocks, as described in what follows.

**User Access and Scheduling:** It is not possible to login and run live experiments from a node. Instead, all measurements

are managed by a central scheduling system that allows to deploy and control experiments over multiple nodes at the same time. Users access the scheduler through a user interface and public APIs. An AngularJS-based web-portal is available for such purpose, although users can also access the scheduler via a Command Line tool (CLI). In any case, platform users need to obtain a user certificate first, which is the same as per FIRE platforms and experimental federations that align with the Fed4FIRE initiative of the European Commission for a distributed, heterogeneous and large-scale experimental network platform.<sup>8</sup> Indeed, as a consequence, concerning the scheduling APIs, the MONROE scheduling is compatible with other Fed4FIRE compatible interfaces.

**Maintenance and Management:** This block is a subsystem used by the maintenance team to administer and maintain MONROE nodes with ease. It relies on a DB with a friendly interface to check node connectivity status and stats.

**Node:** Measurement nodes are dual-apu2 machines with a Linux Debian based operating system. They include core components (e.g., for handling routing and connectivity, for monitoring the software behavior, etc.) plus a set of Linux Docker<sup>9</sup> containers in which experiments are triggered in isolation. Moreover, some containers run constantly in the background to collect results from “default” experiments, e.g., to collect statistics on bandwidth throughput, delay, and gather metadata such as node GPS coordinates. The main idea behind using containers is that it allows for handy control of multiple complex components. Besides, it enables nippy reconfiguration, which can help to implement other containers to the need of other experiments.

**Back-end:** MONROE incorporates several repositories in the back-end, for maintenance and operation of the platform, for collecting metadata and results and temporarily store user data, and also to integrate and import mPlane data analytics computed on the traffic observed by the MONROE nodes.<sup>10</sup> In particular, default experiment results and metadata are collected in a non-relational database using Apache Cassandra,<sup>11</sup> which follows an experiment-oriented design.

**Visualization and open data:** Metadata and default experiment results can be freely visualized in near-real time.<sup>12</sup> Specifically, MONROE offers a user interface that has been designed to provide a graphical representation of node deployment, their status, as well result collection of selected experiments. Measurement data will be openly released.

As shown in Fig. 3, users can fetch the data generated by their experiments from a temporary repository in the MONROE back-end. This is made available through the very same user interface that allows to schedule experiments.

<sup>8</sup><http://www.fed4fire.eu/>

<sup>9</sup><http://www.docker.com>

<sup>10</sup>mPlane (<http://www.ict-mplane.eu>) is a measurement platform designed for fixed networks in the frame of a project funded by the European Commission, and which has been extended by MONROE to account for the analysis of traffic at mobile nodes.

<sup>11</sup><http://cassandra.apache.org>

<sup>12</sup><http://visual.monroe-system.eu/>

<sup>4</sup>PC Engines, apu2 platform: <https://www.pcengines.ch/apu2.htm>

<sup>5</sup>MC7455 miniPCI express (USB 3.0) modem: <https://www.sierrawireless.com/products-and-solutions/embedded-solutions/products/mc7455/>

<sup>6</sup><https://github.com/MONROE-PROJECT>

<sup>7</sup><https://www.monroe-project.eu/resources/papers/>

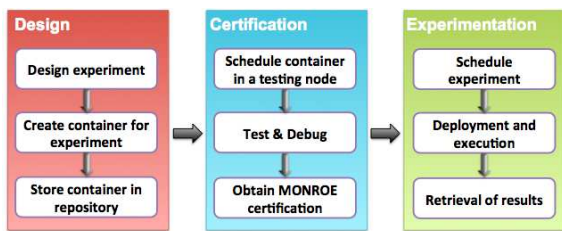


Fig. 4: Experimentation workflow using the MONROE platform.

Remarkably, the measurement structure is not only capable of monitoring and analyzing the network behavior in real-time by scheduling user-defined Docker containers, but also to cache measurement and metadata for offline analysis.

#### B. MONROE Experiment Life-cycle

The workflow of MONROE experiments is depicted in Fig. 4. Initially, experimenters have to specify what kind of measurements they want to do and design them in Docker containers, that includes various software types. This is the design phase depicted in the figure, which includes creating and storing a Docker container in a github repository.

Secondly, containers have to be certified before going live. This is the central block in the figure. Specifically, containers have to be tested and, to this purpose, they can be run in special nodes devoted to debugging, to which users can connect directly and tune the experiments to respect safety and stability rules. In any case, a container has to be eventually certified by the MONROE maintenance team.

Finally, as shown in the rightmost part of the figure, the users deploy their containers into normal nodes through a user interface or the CLI to the scheduler that runs in the backend. The scheduler offers both Fed4FIRE-compatible and REST APIs, and connects to nodes to sense their availability and *health* conditions before making and enforcing any scheduling decision. The user interface, in addition to help to select types, number of nodes, and suitable time-slots, also helps users to collect passive and active results from multiple nodes. Indeed, after successfully finishing the experiments, experimenters can collect their data directly from the user interface along with metadata, e.g., GPS, throughput, and other pieces of background information on the nodes used in the experiment.

#### IV. MEASUREMENT SETUP

The measurement setup we have used consists in a Docker container with a WebRTC streamer and an IP tunnel handler that makes available a multimedia file over HTTPS. The container then includes a light implementation of WebRTC and we have made it publicly available for experimenters willing to design their own tests.<sup>13</sup> When the experiment is scheduled on a set of machines, each of them makes a link available for connecting and watching the multimedia file using a Chrome browser acting as WebRTC client. We connect the WebRTC client from a computer in our lab to a MONROE node using such link. Therefore, the stream goes through the cellular uplink of the MONROE node, crosses the

<sup>13</sup>[docker.monroe-system.eu/deployed/monrtc](https://docker.monroe-system.eu/deployed/monrtc)

TABLE I: MONROE nodes throughput comparison

Node ID	Location	Download [Mb/s]	Upload [Mb/s]
423	Telia SE	0.178-10.38	0.165-1.58
428	Telia SE	0.53-13.97	0.25-2.4
429	Telenor SE	0.78-12.178	0.22-1.74
501	Telenor SE	36.43-68.12	14.93-22.32
591	Vodafone ES	24.58-72.64	17.36-23.14
596	Vodafone IT	32.78-65.41	14.21-25.10

Internet and the access network of our lab, which uses a multi-gigabit connection. Hence, the bottleneck of the stream is in the cellular connectivity experienced by the MONROE node.

The MONROE platform provided us with many static and mobile nodes. Specifically, for the experiments reported in this paper, we have used the nodes listed in Table I. The table includes some limited yet important information about the location of the node. Indeed, we only report operator name and country, whereas more detailed pieces of information about the locations are omitted since this study does not serve as a thorough survey of operator’s performance comparison. The examples of results reported here are only meant to highlight how the potentials of WebRTC have not been fully unleashed by the sub-optimal MBB service available in many places in Europe as of today. However, as we will see later, MBB operators already offer WebRTC-ready connectivity for static users, while mobility is still a big issue.

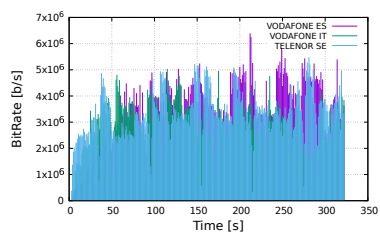
The WebRTC throughput offered by MONROE nodes—i.e., the one offered by the MBB operator used at the node—differs a lot across space and operators, as well as across time, as shown in the third and fourth columns of Table I. Such throughput values have been collected with capacity experiment run a few minutes before and after running the WebRTC experiments.

To generate the figures reported later, we have used the data collected by Google Chrome on peer-to-peer connections, which include WebRTC streams. The resulting logs contain, per each individual stream, the timing and headers of packets received as well as the timing of various internal events such as received frames, losses, bitrate, delay and jitter.

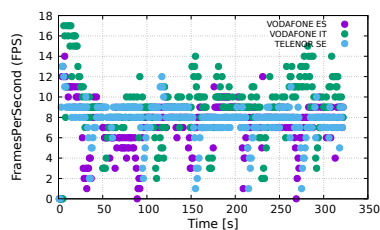
#### V. RESULTS AND OBSERVATIONS

In this section we report statistics on WebRTC performance attributes, as observed by means of Chrome internals at the destination of the WebRTC streams. We run WebRTC streamers in static nodes across Italy, Spain and Sweden, and on mobile nodes in Sweden. We only report a small yet indicative subset of the results we have collected. For each experiment, we plot the following statistics: (i) BitRate, (ii) frames per second (FPS) rendered at the receiver, (iii) packet delay and (iv) jitter. We compare static and mobile cases.

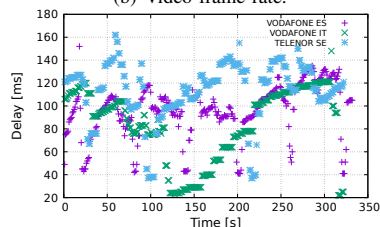
**Static case.** In the first scenario that we consider, the streamer and the client have high quality connectivity (nodes 501, 591, and 596 in Table I). The selected MONROE nodes were connected to various 4G operators and, at the other end of the connection, we use a computer connected to a gigabit LAN directly connected to a multi-gigabit metropolitan network. In most of the test cases, the media stream turned out to be smooth, the final user having a rather good experience



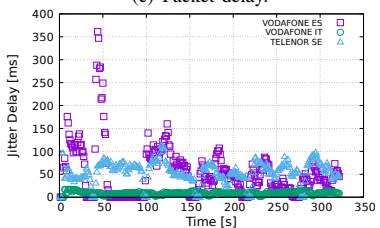
(a) Video bitrate.



(b) Video frame rate.



(c) Packet delay.

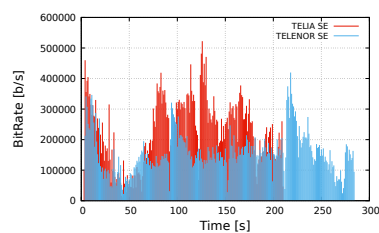


(d) Jitter delay.

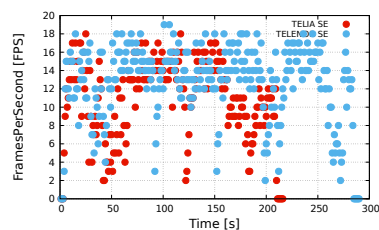
Fig. 5: WebRTC performance experienced by static nodes under different operators in different countries.

with typical bitrates of a few Mb/s, frame rates often above ten frames per second and latency in the ballpark of 100 ms, which is below the threshold of human perception. Jitter was normally below 100 ms. Fig. 5 illustrates an example of performance figures over time for three simultaneous connections using MONROE nodes in three countries. Of course each connection shows different performance figures, and in particular the Spanish sample shows the worst behavior in terms of rate and delay/jitter, although we need to mention that the selected MONROE node in Spain was using an Italian SIM card operating in roaming. In any case, the observed performance is acceptable.

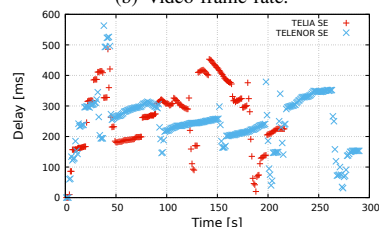
**Mobile case.** In this scenario we select MONROE nodes in buses in Sweden (nodes 423, 428, and 429 in Table I), while the other end of the connection is in our lab, as in the other case. Fig. 6 shows that the user experience was not as smooth as in the first case since the bitrate was significantly degraded. Although WebRTC was able to keep the stream going, frame



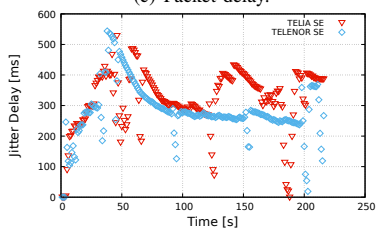
(a) Video bitrate.



(b) Video frame rate.



(c) Packet delay.

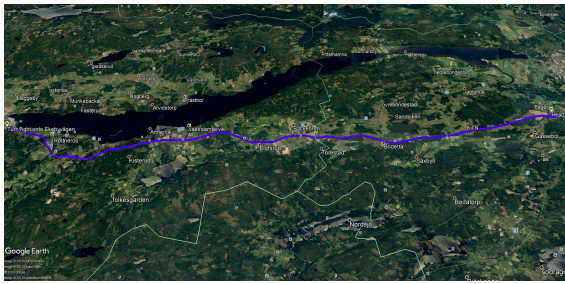


(d) Jitter delay.

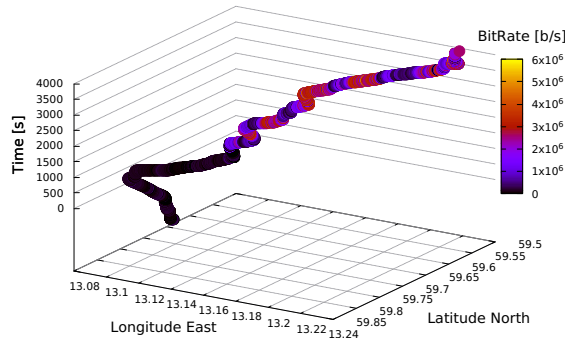
Fig. 6: WebRTC performance figures observed for a bus in a medium-size city in Sweden.

rate was very variable, and delay and jitter were annoyingly high in all locations and for all operators. We have observed many other cases like this in our measurements (and not only in Sweden, of course), which leads to the conclusion that current MBB networks are not ready to fully support WebRTC (and alike multimedia services) on the move.

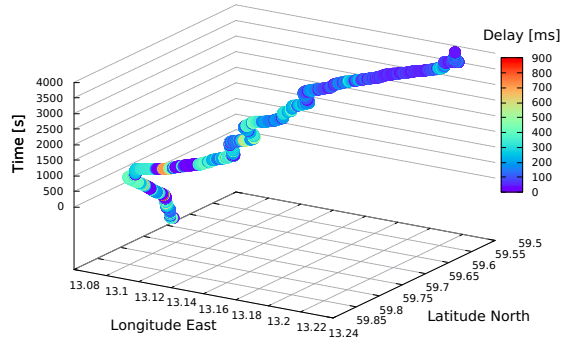
To further understand the behavior of WebRTC on MBB networks, Fig. 7 shows 3D plots of bitrate and delay experienced by a MONROE node on a bus, for one of the operators in Sweden, as a function of the geographical position of the node (also shown in the topmost subfigure using Google Earth). The information about coordinates is provided by the MONROE node itself, within a metadata stream generated during the experiments. From the figure it is clear that some areas crossed by the bus were very poorly served, so that delay are constantly high and bitrates change smoothly over the trajectory, indicating that coverage (signal strength) is far from optimal in that area.



(a) Bus route.



(b) Bitrate.



(c) Packet delay.

Fig. 7: WebRTC video performance measured at destination, on a public bus on service in a medium-size city in Sweden.

From the above results, it is clear that peer-to-peer communication over the Internet is still not “ideal” and whether one gets good QoE or not for the whole communication period completely depends on how good the network connection is, and how good it remains across the whole time of the communication session. In the ideal WebRTC scenario, endpoints are web browsers running on reasonably powerful laptops with strong WiFi or wired network connections, communicating on top of a reasonably consistent network. This should work well. However, if the devices are mobile and have a non-consistent and often not good wireless connection, then the quality of the communication is likely to be below any acceptable threshold, as observed in the mobile case described above.

## VI. CONCLUSIONS

In this work, we have evaluated the performance of WebRTC for static and mobile users by leveraging the MONROE

platform. To this aim, we have designed an open tool, running in a Docker container, for generating WebRTC sessions in mobile nodes by using standard WebRTC APIs. As such, the work presented a complete and novel methodology for the performance evaluation of web services using operational MBB networks. As an initial result, we have observed that mobility is still an important challenge for WebRTC, since MBB operators do not yet provide with full quality coverage for users on the move. Our approach can be used in the future for a broader and continuous assessment of WebRTC.

## ACKNOWLEDGMENTS

Work funded by the EU H2020 research and innovation programme under grant agreement No. 644399 (MONROE). The work of V. Mancuso was supported by the Ramon y Cajal grant (ref: RYC-2014-16285) from the Spanish Ministry of Economy and Competitiveness.

## REFERENCES

- [1] A. Zeidan, A. Lehmann, and U. Trick, “WebRTC enabled multimedia,” in *proceedings of World Telecommunications Congress 2014*, Jun. 2014.
- [2] A. Johnston, J. Yoakum, and K. Singh, “Taking on WebRTC in an enterprise,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 48–54, April 2013.
- [3] S. Loreto and S. P. Romano, “How Far Are We from WebRTC-1.0? An Update on Standards and a Look at What’s Next,” *IEEE Communications Magazine*, vol. 55, no. 7, pp. 200–207, 2017.
- [4] FCC, “2013 Measuring Broadband America February Report,” FCC’s Office of Engineering and Technology and Consumer and Governmental Affairs Bureau, Tech. Rep., 2013.
- [5] O. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunström, A. Safari Khatouni, M. Mellia, and M. Ajmone Marsan, “Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks,” in *Proc. of ACM Mobicom.*, 2017.
- [6] B. Garcia, F. Gortazar, L. Lopez-Fernandez, M. Gallego, and M. Paris, “WebRTC Testing: Challenges and Practical Solutions,” *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 36–42, 2017.
- [7] B. Grozev, G. Politis, E. Iovov, T. Noel, and V. Singh, “Experimental Evaluation of Simulcast for WebRTC,” *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 52–59, 2017.
- [8] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, “A First Look at Cellular Network Performance during Crowded Events,” in *Proc. of SIGMETRICS*, 2013.
- [9] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z.-M. Mao, S. Sen, and O. Spatscheck, “An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance,” in *Proc. of SIGCOMM*, 2013.
- [10] Tektronix, “Reduce Drive Test Costs and Increase Effectiveness of 3G Network Optimization,” Tektronix Comm., Tech. Rep., 2009.
- [11] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, “Netalyzr: Illuminating the edge network,” in *Proc. of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 246–259.
- [12] N. Vallina-Rodriguez, “Illuminating the Third Party Mobile Ecosystem with the Lumen Privacy Monitor,” in *FTC PrivacyCon 2017*, January 2017.
- [13] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, “On the seamless interaction between WebRTC browsers and SIP-based conferencing systems,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 42–47, April 2013.
- [14] E. Bertin, S. Cubaud, S. Tuffin, N. Crespi, and V. Beltran, “WebRTC, the day after: What’s next for conversational services?” in *2013 17th International Conference on Intelligence in Next Generation Networks (ICIN)*, Oct 2013, pp. 46–52.
- [15] A. Amirante, T. Castaldi, A. Gouaillard, L. Miniero, S. G. Murillo, and S. P. Romano, “Bringing privacy to the Janus WebRTC server: The PERC way,” in *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, Sept 2017, pp. 1–8.